Java Object-Oriented Programming

- Giảng viên : Nguyễn Đức Hiển
  - Email : ndhien@udn.vn
  - Website :

- Thời lượng
  - Lý thuyết : 2 tín chỉ (30 tiết)
  - Thực hành + thảo luận : 1 tín chỉ

Lập trình cơ sở dữ liệu JDBC

(Java DataBase Connectivity)

Nội dung

- Giới thiệu
- Kết nối và truy xuất cơ sở dữ liệu
- Xử lý kết quả vấn tin

Giới thiệu về JDBC

- JDBC (Java DataBase Connectivity) là một thư việc chuẩn dùng để truy xuất các cơ sở dữ liệu như MS Acess, SQL Server, Oracle,… trong các ứng dụng Java bằng ngôn ngữ truy vấn SQL.
- Các hàm truy xuất cơ sở dữ liệu với JDBC nằm trong gói java.sql.*
- JDBC driver: JDBC bao gồm hai phần:
  - JDBC API: là một API hoàn toàn dựa trên Java.
  - JDBC DriverManager: là trình quản lý JDBC giao tiếp trực tiếp với các trình điều khiển cơ sở dữ liệu cụ thể - giao tiếp thực sự với cơ sở dữ liệu.

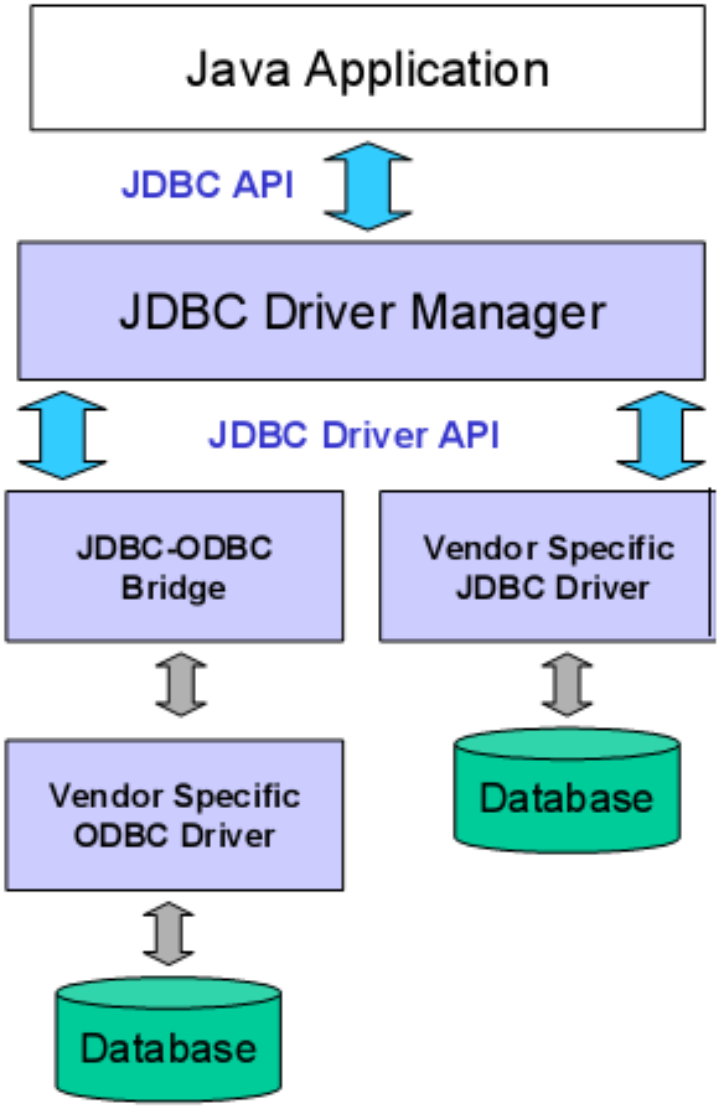
JDBC Database Driver

- Kiểu 1: JDBC-ODBC bridge driver
  - Chuyển đổi các lời gọi JDBC thành ODBC, ODBC có thể truy xuất giao thức DBMS.
  - Phương thức truy xuất dữ liệu đòi hỏi trình điều khiển ODBC được cài đặt trên máy tính client.
- Kiểu 2: Native protocol partly java driver
  - Chuyển lời gọi JDBC thành các lời gọi giao thức DBMS đặc thù.
  - Khi đó sự chuyển đỗi này đặt trên máy client, một số mã nhị phân phải được cài đặt trên máy tính client.

JDBC Database Driver

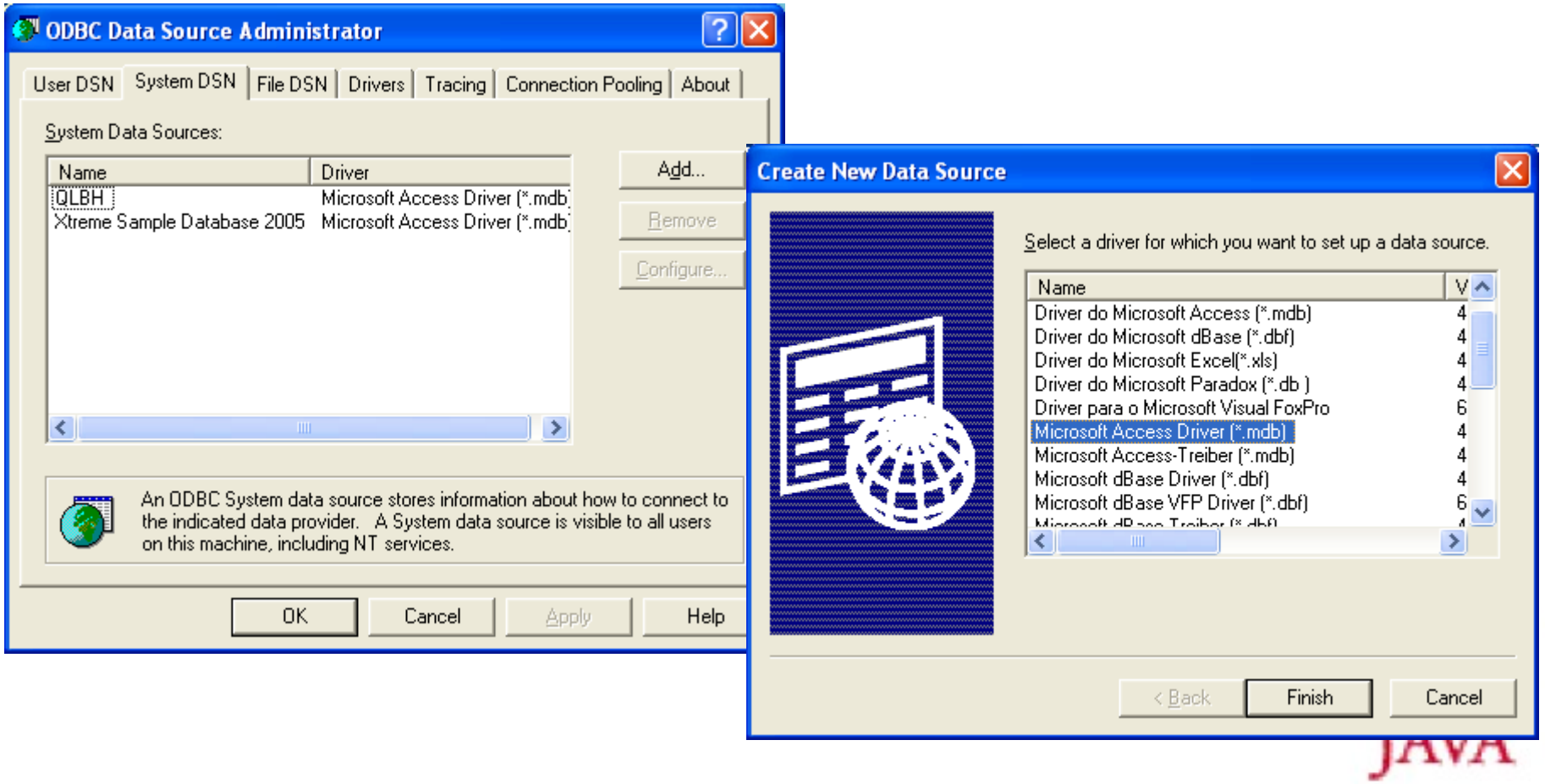
- Kiểu 3: Net protocol all Java driver
  - Chuyển đổi các lời gọi JDBC thành giao thức mạng độc lập với bất kỳ giao thức DBMS đặc thù. Sau đó, một phần mềm trung gian (middleware) chạy trên máy server chuyển đỗi giao thức mạng thành giao thức DBMS đặc thù.
  - Sự chuyển này đặt ở phía server mà không đòi hỏi cài đặt trên máy tính client.
- Kiểu 4: Native protocol all Java driver
  - Chuyển lời gọi JDBC thành các lời gọi giao thức DBMS đặc thù.
  - Khi đó sự chuyển đỗi này đặt phía server, mà không đòi hỏi cài đặt trên máy tính client

Cơ chế hoạt động với JDBC



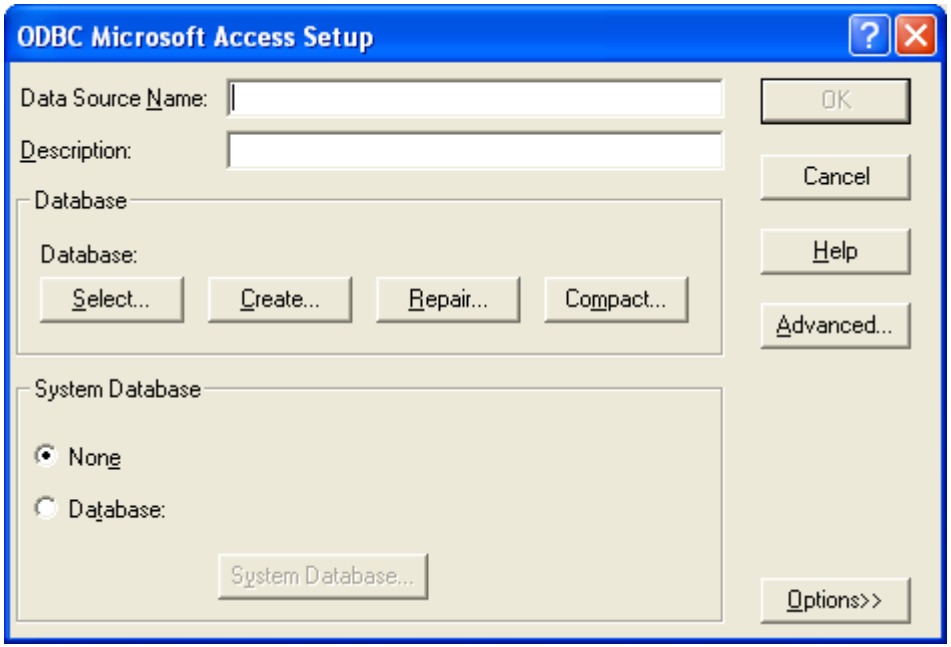
Tạo nguồn dữ liệu ODBC

- Trên Window, vào Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC)



Tạo nguồn dữ liệu ODBC

- Đặt tên nguồn dữ liệu ở mục “Data Source Name” (sẽ sử dụng trong chuỗi kết nối)
- Nhấp “Select” để chọn đường dẫn đến file cơ sơ dữ liệu.



Các bước truy xuất CSDL

- Nạp trình điều khiển.
- Thiết lập kết nối.
- Tạo đối tượng Statement
- Thực hiện vấn tin
- Xử lý kết quả trả về
- Đóng kết nối

Nạp trình điều khiển

- Sử dụng phương thức tĩnh forName() của lớp Class với tham số là tên trình điều khiển cơ sở dữ liệu.
- Cách dùng:

```
try {
    Class.forName("Database driver name");
}
catch ( ClassNotFoundException e){
    System.out.println("Driver not found : " +
        e.getMessage());
}
catch ( SQLException e){
    System.out.println("SQL Exception : " +
        e.getMessage());
}
```

Nạp trình điều khiển

- Trình điều khiển của MySQL:
  - Class.forName(“org.gjf.mm.mysql.Driver”);
- Trình điều khiển của Oracle:
  - Class.forName(“oracle.jdbc.driver.OracleDriver”);
- Trình điều khiển của Sybase:
  - Class.forName("com.sybase.jdbc.SybDriver");
- Trình điều khiển qua cầu nối ODBC:
  - Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);

Định nghĩa chuỗi kết nối

```
String host = "dbhost.yourcompany.com";
String dbName = "someName";
int port = 1234;
String oracleURL = "jdbc:oracle:thin:@" + host +
    ":" + port + ":" + dbName;
String sybaseURL = "jdbc:sybase:Tds:" + host  +
    ":" + port + ":" + "?SERVICEid=" + dbName;
```

Thiết lập kết nối

- Để thiết lập kết nối ta gọi phương thức tĩnh getConnection() của lớp DriverManager, khi đó trả về một thể hiện của lớp Connection, theo dạng như sau:
  - `String user = “sa”`
  - `String password = “secret”`
  - `Connection con = DriverManager.getConnection(dbUrl, username, password);`
- Trong đó:
  - **`dbUrl`**: là chuổi kết nối đến cơ sở dữ liệu.
  - **`username`** : tên người dùng đăng nhập
  - **`password`** : mật khẩu đăng nhập.

Thông tin cơ sở dữ liệu

- Để lấy các thông tin về cơ sở dữ liệu gọi phương thức getMetaData() của Connection trả về đối tượng lớp DatabaseMetaData.
- Ví dụ:

```
DatabaseMetaData dbMetaData = connection.getMetaData();

String productName =
    dbMetaData.getDatabaseProductName();
System.out.println("Database: " + productName);

String productVersion =
    dbMetaData.getDatabaseProductVersion();
System.out.println("Version: " + productVersion);
```

Ví dụ kết nối đến MySQL

```
Connection connection = null;
try {
    Class.forName("org.gjt.mm.mysql.Driver");
    String dbUrl= "jdbc:mysql://localhost/qlbh";
    String username = "root";
    String password = "";
    connection = DriverManager.getConnection(dbUrl,
        username, password);
}
catch(ClassNotFoundException e){
  System.out.println("Driver not found : " + e.getMessage())
}
catch(SQLException e){
    System.out.println("SQL Exception : " + e.getMessage());
}
```

Ví dụ cách kết nối đến Oracle

```
Connection connection = null;
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    String dbUrl= "jdbc:oracle:thin@localhost/qlbh";
    String username = "scott";
    String password = "tiger";
    connection = DriverManager.getConnection(dbUrl,
        username, password);
}
catch(ClassNotFoundException e){
  System.out.println("Driver not found : " + e.getMessage());
}
catch(SQLException e){
    System.out.println("SQL Exception : " + e.getMessage());
}
```

Ví dụ cách kết nối qua ODBC

```
Connection connection = null;
try {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   String dbURL = "jdbc:odbc:qlbh";
   String username = "sa";
   String password = "";
   connection = DriverManager.getConnection(dbURL,
      username, password);
}
catch(ClassNotFoundException e){
   System.out.println("Driver not found : " + e.getMessage());
}
catch(SQLException e){
    System.out.println("SQL Exception : " + e.getMessage());
}
```

Tạo đối tượng Statement

- Một đối tượng Statement được sử dụng để truyền câu truy vấn và câu lệnh đến CSDL và nó được tạo từ đối tượng Connection đã khởi tạo.
- Cách tạo đối tượng statement, gọi phương thức createStatement() của đối tượng Connection:
  - `Statement statement = connection.createStatement();`
- Theo mặc đinh, đối tượng statement được tạo ra từ phương thức createStatement() khi truy vấn đến cơ sở dữ liệu cho kết quả là forward-only và read-only.

Tạo đối tượng Statement

- Tạo statement cho phép cập nhật:
  - createStatement(int resultSetType,int resultSetConcurrency) throws SQLException
- Cho phép tạo đối tượng Statement mà sẽ phát sinh đối tượng ResultSet với kiểu và thao tác xác định.
- Các tham số:
  - resultSetType: kiểu của tập kết quả, có thể là ResultSet.TYPE_FORWARD_ONLY, ResultSet.TYPE_SCROLL_INSENSITIVE, hoặc ResultSet.TYPE_SCROLL_SENSITIVE
  - resultSetConcurrency: kiểu thao tác, có thể là ResultSet.CONCUR_READ_ONLY hoặc ResultSet.CONCUR_UPDATABLE

Thực hiện vấn tin

- Xây dựng câu truy vấn (query):
  - `String sql = "SELECT col1, col2,col3 FROM table1, table2";`
  - `String sql = "Update table1 set col1 = col1 *10 / 100";`
- Để thực thi câu lệnh SQL ta sử dụng một trong hai phương thức sau của đối tượng Statement là executeQuery() và executeUpdate().
  - Phương thức executeQuery: thực hiện câu vấn tin dạng SELECT và nhận kết quả trả về là một đối tượng ResultSet (tập các bản ghi dữ liệu truy vấn được).
  - Phương thức executeUpdate: thực thi câu vấn tin dạng CREATE, UPDATE, INSERT, DELETE.

Xử lý kết quả trả về (ResultSet)

- Đối với câu vấn tin dạng SELECT, nếu truy vấn thành công thì kết quả trả về là tập các bản ghi dữ liệu được lưu trong đối tượng ResultSet và ta có thể hiển thị hoặc xử lý trên kết quả này.
- Cách duyệt qua các bản ghi dữ liệu như sau:

```
Statement stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery(query);
while( rs.next()){
    out.println(rs.getString(1));
    out.println(rs.getString(2));
    //...
}
```

Đối tượng ResultSet

- Theo mặc định phương thức createStatement() tạo ra đối tượng ResultSet là forward-only & Read-only. Điều này có nghĩa ta chỉ có thể di chuyển con trỏ thông qua nó từ bản ghi đầu tiên đến bản ghi cuối cùng mà không thể cập nhật được.
- Khi ResultSet được tạo, con trỏ được định vị trước bản ghi đầu tiên. Sau đó ta có thể sử dụng các phương thức của ResultSet để di chuyển con trỏ.
- Các phương thức của ResultSet ném ra ngoại lệ SQLException.

Đối tượng ResultSet

- Ví dụ tạo ResultSet gồm 1 dòng và 1 cột:
  - `Statement stmt = connection.createStatement();`
  - `ResultSet rsUserID = stmt.executeQuery("SELECT UserID FROM User WHERE Email='jerry@yahoo.com'");`
- Ví dụ ResultSet gồm nhiều dòng và nhiều cột:
  - `Statement stmt = connection.createStatement();`
  - `ResultSet rsProducts = stmt.executeQuery("SELECT * FROM product");`

Đối tượng ResultSet

- Một số phương thức của ResultSet forward-only, read-only:

Method	Description
`next()`	Moves the cursor to the next row in the result set.
`last()`	Moves the cursor to the last row in the result set.
`close()`	Releases the result set's JDBC and database resources.
`getRow()`	Returns an int value that identifies the current row of the result set.

Cách nhận dữ liệu từ ResultSet

- Các phương thức dạng getXXX() có thể sử dụng để lấy về các kiểu dữ liệu cơ bản, như int, long, double, …
  - `int n = rsProduct.getInt(“Soluong”),`
  - `double price = rsProduct.getDouble(“Price”);`
  - `...`
- Các phương thức dạng getXXX() cũng có thể được sử dụng để lấy về kiểu chuỗi, ngày tháng, thời gian, …
  - `String username = rsUser.getString(1);`
  - `Date = rsProduct.getDate(“MakeDate”);`
  -

Cập nhật dữ liệu

- Thêm một bản ghi mới vào cơ sở dữ liệu
- Ví dụ:

```
String query =
    "INSERT INTO Products(product_id, " +
    " product_name, product_price, unit) " +
    "VALUES ('" + id + "', '" + name + "', '" +
    price + "', '" + unit + "')";

Statement statement = connection.createStatement();
int rowCount = statement.executeUpdate(query);
```

Cập nhật dữ liệu

- Cập nhật các thay đổi vào cơ sở dữ liệu
- Ví dụ:

```
String query = "UPDATE Products SET " +
    "product_name = '" + name + "', " +
    "product_price = '" + price + "', " +
    "unit = '" + unit + "' " +
    "WHERE product_id = '" + id + "'";

Statement statement = connection.createStatement();
int rowCount = statement.executeUpdate(query);
```

Cập nhật dữ liệu

- Xóa các bản ghi thỏa mãn điều kiện
- Ví dụ:

```
String query = "DELETE FROME Products " +
    "WHERE product_id = '" + id + "'";

Statement statement = connection.createStatement();
int rowCount = statement.executeUpdate(query);
```

Sử dụng PreparedStatement

- Là cách biểu diễn các câu SQL dưới dạng các tham số và câu lệnh được biên dịch trước.
- Ví du:

```
String preparedSQL = "SELECT * " + "FROM Products " +
    "WHERE product_id = ?";
PreparedStatement pstmt =
    connection.prepareStatement(preparedSQL);

String id = "A05"
pstmt.setString(1, id);
ResultSet product = pstmt.executeQuery();
```

Sử dụng PreparedStatement

- Khi sử dụng PreparedStatement, database server phải kiểm tra cú pháp và chuẩn bị một kế hoạch xử lý một lần cho mỗi câu lệnh SQL.
- Để xác định một tham số cho PreparedStatement, gỏ dấu chấm hỏi (?) trong câu lệnh SQL.
- Để cung cấp các giá trị cho các tham số trong PreparedStatement, sử dụng các phương thức dạng setXXX().
- Để xử lý câu SELECT, sử dụng phương thức executeQuery(). Để xử lý câu INSERT, UPDATE, hoặc DELETE, sử dụng phương thức executeUpdate().

Sử dụng PreparedStatement

- Sử dụng PreparedStatement để sữa đổi dữ liệu
- Ví dụ:

```
String preparedSQL = "UPDATE Products SET " +
    " product_name = ?, " + " price = ?" +
    "WHERE product_id = ?";

PreparedStatement ps =
    connection.prepareStatement(preparedSQL);
ps.setString(1, "Laptop IBM Lenovo");
ps.setString(2, "$1204");
ps.setString(3, "A05");
```

Sử dụng PreparedStatement

- Sử dụng PreparedStatement thêm mới một bản ghi
- Ví dụ:

```
String preparedQuery = "INSERT INTO Products " +
    "(product_id, product_name, price) " +
    "VALUES (?, ?, ?)";

PreparedStatement ps =
    connection.prepareStatement(preparedQuery);
ps.setString(1, "TO4");
ps.setString(2, "HP Printer 3400");
ps.setDouble(3, "$120");

ps.executeUpdate();
```

Sử dụng PreparedStatement

- Sử dụng PreparedStatement xóa các bản ghi thõa mãn điều kiện.
- Ví dụ:

```
String preparedQuery = "DELETE FROM Products " +
    "WHERE product_id = ?";
PreparedStatement ps =
    connection.prepareStatement(preparedQuery);

ps.setString(1, "AO5");
ps.executeUpdate();
```

Ngắt kết nối và giải phóng tài nguyên

- Để ngắt kết nối ta sử dụng phương thức close của đối tượng connection:
  - connetion.close()

Sample 1

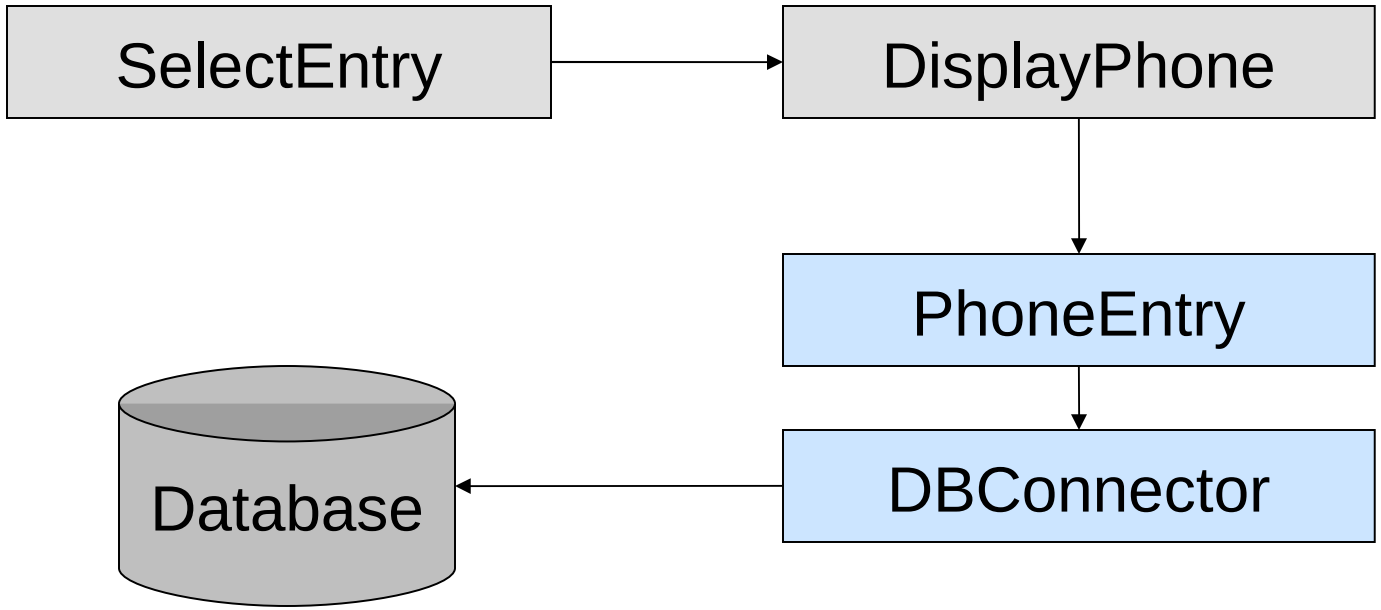
```
import java.sql.*;
public class SalesData {
  public static void main(String[] args) {
     String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
     Statement statement = null;
     ResultSet resultset = null;
     try {
        Class.forName(driver);
        Connection connection =
           DriverManager.getConnection("jdbc:odbc:Sales");
        statement = connection.createStatement();
        String query = "SELECT * FROM dmhang";
        resultset = statement.executeQuery (query);
        System.out.println("\tID\tDesc\tQuantity\tPrice");
```

Sample 1 (Cont.)

```
        while(rs.next()){
            System.out.print( "\t" + resultset.getString(1));
            System.out.print( "\t" + resultset.getString(2));
            System.out.print( "\t" + resultset.getString(3));
            System.out.println( "\t" + resultset.getString(4));
        }
    }
    catch ( ClassNotFoundException e) {
        System.out.println("No such class : " + e.getMessage());
    }
    catch ( SQLException e) {
        System.out.println("SQL Exception : " + e.getMessage());
    }
```

“Phones” Application

- Một ứng dụng đơn giản hiển hiển thị các bản ghi trong cơ sở dữ liệu.
- Kiến trúc:



Thanks for listenning!!!